Faculté des Sciences et Ingénierie - Sorbonne Université

Master Informatique parcours ANDROIDE

IAR - Intelligence Artificielle pour la Robotique

TME 1

Tuning TD3 for LunarLanderContinuous-v2

Réalisé par :
Joe CHAMOUN - 21312860
Walid GHENAIET - 21307720

GitHub repo : `https://github.com/JoeCham3oun/IAR_TME1_TD3`

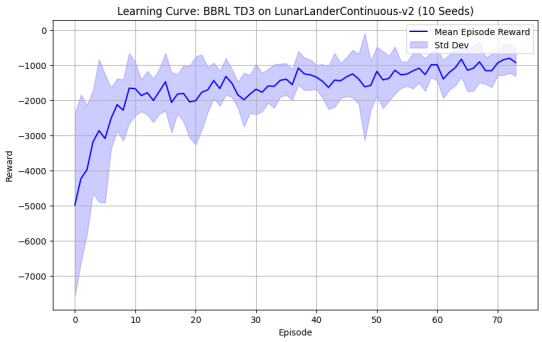
Octobre 2024

Table des matières

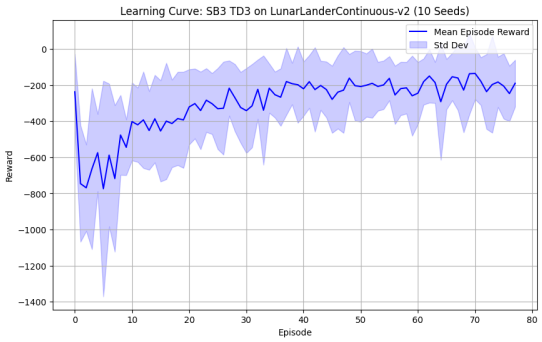
Chapitre 1

Implémentation Personnalisée vs Stable Baselines3

Dans cette analyse, nous comparons deux courbes de récompenses obtenues en utilisant l'algorithme TD3, exécuté avec les mêmes hyper-paramètres et les mêmes seeds sur l'environnement *LunarLanderContinuous-v2*. Cependant, la première courbe représente notre propre implémentation de TD3, tandis que la seconde courbe est générée à partir de l'implémentation de TD3 fournie par la bibliothèque *Stable Baselines3 (SB3)*.



(a) Performance de notre implémentation TD3



(b) Performance de TD3 de Stable Baselines3

Figure 1.1 – Comparaison des courbes d'apprentissage TD3 : Implémentation personnalisée vs Stable Baselines3

Dans la **Figure 1.1a**, qui représente notre implémentation de TD3, nous observons une amélioration plus régulière des performances. Après un départ difficile avec des récompenses chutant autour de -5000, les performances s'améliorent considérablement dès l'épisode 10. La courbe montre ensuite une progression stable, atteignant environ -1000 vers la fin de l'entraînement après 70 évaluations. L'agent montre une capacité d'apprentissage solide avec des fluctuations réduites vers la fin.

Dans la **Figure 1.1b**, obtenue avec la version SB3 de TD3, les résultats sont également impressionnants. Après un début plus stable que notre implémentation, les récompenses oscillent beaucoup moins et s'améliorent plus rapidement, atteignant des scores positifs proches de 200 dès l'épisode 70. La performance générale montre une meilleure stabilité et des récompenses supérieures en moyenne par rapport à notre implémentation.

1.1 Analyse comparative

— Les deux implémentations montrent une tendance à l'amélioration de l'agent sur *LunarLanderContinuous-v2*. Cependant, l'implémentation SB3 de TD3 reste plus performante que notre implémentation. Non seulement elle atteint des scores positifs plus rapidement, mais elle montre également moins de fluctuations importantes par rapport à notre version, bien que notre implémentation se soit grandement améliorée dans cette version.

— La première courbe (Figure 1.1a) montre que notre agent apprend efficacement après un démarrage lent, avec une amélioration significative autour de l'épisode 10. Toutefois, la version SB3 (Figure 1.1b) atteint des scores nettement supérieurs, ce qui montre un apprentissage plus rapide et plus stable.

En conclusion, bien que notre implémentation TD3 ait montré des progrès notables, l'implémentation SB3 continue de surpasser la nôtre en termes de stabilité et de rapidité d'apprentissage, atteignant des performances optimales avec des fluctuations moindres.

1.2 Comparaison entre BBRL TD3 et SB3 TD3

Après avoir analysé les courbes de récompense des deux implémentations (voir Figure 1.1), nous avons observé une différence de performance particulièrement significative, avec SB3 TD3 démontrant des résultats supérieurs. Bien que nous ayons calculé les récompenses moyennes sur plusieurs graines pour tirer cette conclusion, il est important de reconnaître que cette disparité pourrait encore être influencée par la chance. Par conséquent, un test statistique est nécessaire pour confirmer si SB3 TD3 est véritablement supérieur.

1.3 Analyse Statistique des Différences de Performance

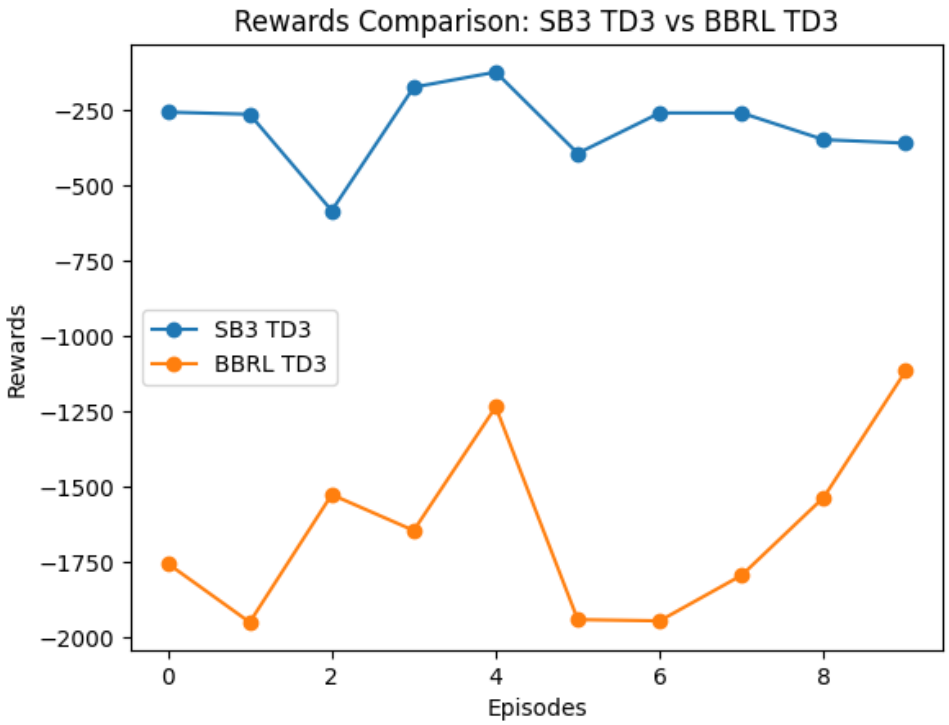


FIGURE 1.2 – Courbes de récompense moyennes par seed pour les implémentations BBRL TD3 et SB3 TD3

Pour évaluer quantitativement la différence de performance entre notre BBRL TD3 et SB3 TD3, nous avons effectué un test t sur les récompenses finales sur 10 graines.

1.3.1 Configuration du Test T

— **Hypothèse Nulle (H0)** : Il n'y a pas de différence significative entre les récompenses finales de BBRL TD3 et SB3 TD3.

— **Hypothèse Alternative (H1)** : Il y a une différence significative entre les récompenses finales des deux implémentations.

1.3.2 Résultats du Test T

— **T-statistique** : 13.15

— **P-valeur** : 3.52e-07

La p-valeur obtenue est largement inférieure au seuil conventionnel de 0,05, ce qui nous conduit à rejeter l'hypothèse nulle. Ce résultat indique une différence statistiquement significative de performance entre les deux implémentations. La forte t-statistique

et la très faible p-valeur suggèrent que la disparité observée dans la performance est peu susceptible d'être due au hasard et est plutôt attribuable aux différences inhérentes dans les implémentations elles-mêmes.

Étant donné ces résultats, un examen plus approfondi du code est justifié pour identifier les facteurs spécifiques contribuant à cet écart de performance.

1.4 Optimisation des Hyper-paramètres

Dans notre effort pour rapprocher les performances de notre implémentation TD3 de celles obtenues avec *Stable Baselines3 (SB3)*, nous avons décidé de modifier plusieurs hyper-paramètres clés.

- **Nombre de pas par mise à jour (*n_steps*)**
  - *Valeur initiale* : 1
  - *Nouvelle valeur* : 4
  - *Raison du changement* : L'augmentation de *n_steps* à 4 permet à l'agent d'accumuler plus de transitions avant de mettre à jour les réseaux de politiques et de critiques. Cela améliore la qualité des données utilisées pour chaque mise à jour, rendant ainsi l'apprentissage plus robuste.
- **Learning Starts**
  - *Valeur initiale* : 100
  - *Nouvelle valeur* : 15 000
  - *Raison du changement* : En retardant le début de l'apprentissage jusqu'à ce que 15 000 transitions aient été collectées, l'agent peut s'entraîner sur des données plus diversifiées, ce qui réduit le risque d'apprendre prématurément sur des échantillons peu représentatifs, améliorant ainsi la stabilité des premières mises à jour.
- **Taille du Buffer (*buffer_size*)**
  - *Valeur initiale* : 1 000 000
  - *Nouvelle valeur* : 2 000 000
  - *Raison du changement* : En doublant la taille du buffer, l'agent dispose d'une plus grande diversité de transitions historiques sur lesquelles s'entraîner, ce qui peut améliorer la performance globale en permettant une réutilisation plus efficace des expériences passées.

— **Action Noise**
  — *Valeur initiale* : 0
  — *Nouvelle valeur* : 0.1
  — *Raison du changement* : L'ajout d'un bruit d'action avec une valeur de 0.1 favorise une exploration plus agressive de l'espace des actions, aidant ainsi l'agent à découvrir de meilleures politiques et à éviter les minima locaux sous-optimaux.

— **Taux d'Apprentissage (*learning rate*)**
  — *Valeur initiale* : 1e-3 pour l'acteur et le critique
  — *Nouvelle valeur* : 2.5e-4 pour l'acteur et le critique
  — *Raison du changement* : En diminuant le taux d'apprentissage, nous espérons obtenir des mises à jour plus fines et plus stables, ce qui permet une convergence plus douce et évite des fluctuations importantes dans les mises à jour des paramètres.

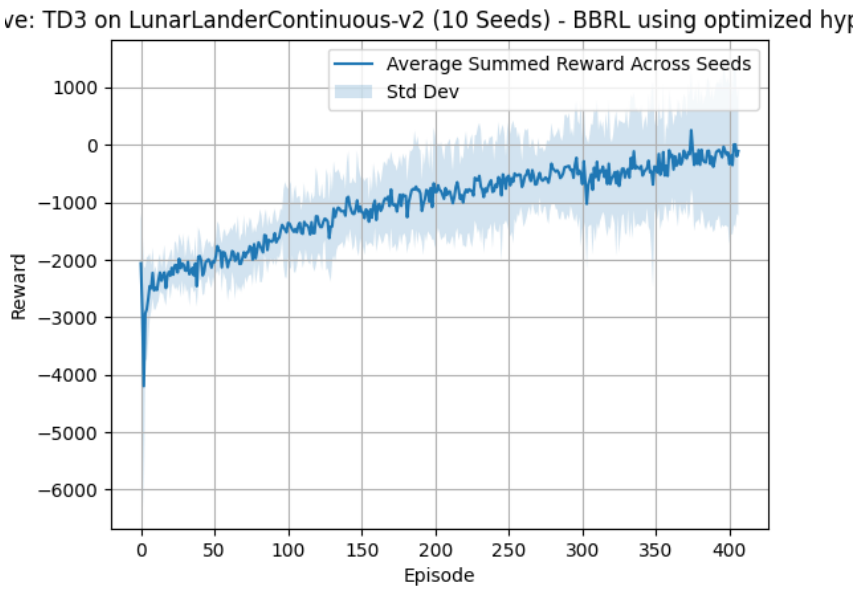


FIGURE 1.3 – Courbe TD3 avec des hyperparamètres optimisés sur l'environnement LunarLanderContinuous-v2.

1.4.1 Analyse des Performances

La **Figure 1.3** montre l'évolution des récompenses moyennes pour notre implémentation TD3 utilisant des hyper-paramètres optimisés. On observe une amélioration progressive et stable des performances au fil des épisodes. L'agent passe d'une récompense initiale très basse (environ -5000) à des récompenses positives après l'épisode 300.

La courbe suggère que l'agent apprend de manière plus régulière, bien qu'à un rythme plus lent par rapport à SB3. Les fluctuations de récompenses sont présentes mais se stabilisent vers la fin, atteignant une moyenne positive après environ 350 épisodes. Cela montre que l'agent parvient finalement à mieux explorer l'environnement et à effectuer des atterrissages plus efficaces, ce qui se traduit par des récompenses positives.

1.4.2 Comparaison avec SB3

En comparant ces résultats avec ceux de l'implémentation SB3 (**Figure 1.1b**), nous remarquons que SB3 présente une progression beaucoup plus rapide. L'agent utilisant SB3 atteint des récompenses positives autour de l'épisode 70, bien avant notre implémentation optimisée. De plus, la stabilité de la courbe SB3 est notable : les fluctuations sont plus réduites et l'agent atteint des récompenses stables d'environ 200.

Cela montre que SB3 a une meilleure efficacité dans l'apprentissage et l'exploration de l'environnement. Toutefois, notre implémentation optimisée parvient tout de même à rattraper les performances, bien que de manière plus lente et avec plus de variance.

1.4.3 Proximité entre les Hyper-paramètres Optimisés et SB3 Params

Lorsque l'on compare notre implémentation optimisée à celle utilisant les hyper-paramètres de SB3 (**Figure 1.1a**), il est évident que notre version optimisée produit des résultats plus proches des performances de SB3. Les récompenses obtenues avec les hyper-paramètres SB3 montrent une amélioration, mais l'agent reste largement en dessous de 0, atteignant à peine -1000 vers l'épisode 70. En revanche, notre implémentation optimisée parvient à dépasser cette limite et finit par obtenir des récompenses positives.

Les hyper-paramètres optimisés permettent donc de rapprocher notre implémentation des résultats obtenus avec SB3, bien qu'il reste un écart notable en termes de rapidité et de stabilité.

1.4.4 Conclusion

Les résultats montrent que les hyper-paramètres optimisés offrent une meilleure performance par rapport à l'utilisation des hyper-paramètres SB3 dans notre implémentation. Bien que l'implémentation SB3 soit plus rapide et plus stable, notre implémentation avec optimisation atteint finalement des récompenses positives après environ 350 épisodes, se rapprochant ainsi des résultats de SB3.

Cette analyse suggère que les ajustements des hyper-paramètres, notamment *n_steps*,

learning_starts, et *action_noise*, ont permis d'améliorer la capacité d'exploration de l'agent et la stabilité de l'apprentissage, tout en restant plus proche des performances globales de SB3.

1.5 Références

— `https://github.com/DLR-RM/stable-baselines3/tree/master`

— `https://github.com/DLR-RM/stable-baselines3/blob/master/stable_baselines3/td3/td3.py`

Chapitre 2

Implémentation Personnalisée vs Tianshou

2.1 Résultats avec les Hyper-paramètres de Tianshou

Après avoir utilisé les hyper-paramètres de la bibliothèque *Tianshou*, nous avons généré une nouvelle courbe de récompense pour l'agent TD3 sur l'environnement *LunarLanderContinuous-v2*. Cette section compare les performances obtenues avec ces nouveaux hyper-paramètres à celles obtenues précédemment avec notre implémentation et celle de *Stable Baselines3 (SB3)*.

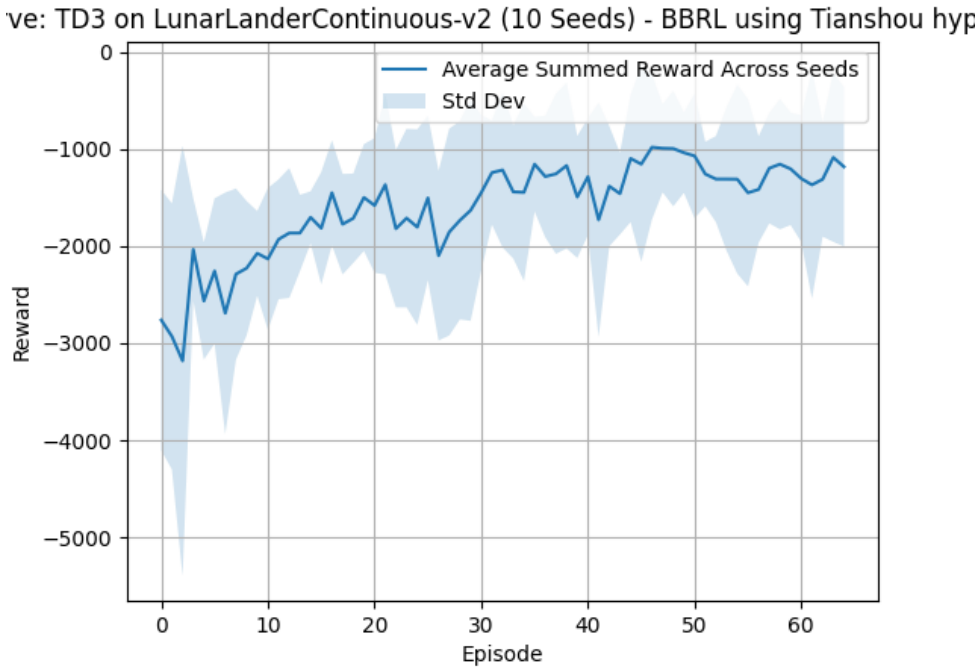


Figure 2.1 – Courbe de récompense moyenne après utilisation des hyper-paramètres de Tianshou

2.1.1 Analyse

L'utilisation des hyper-paramètres de *Tianshou* a permis d'observer certaines différences notables dans les performances de l'agent. La courbe montre les tendances suivantes :

— Une amélioration progressive des performances de l'agent, bien qu'elle reste fluctuante à certains épisodes.

— Une certaine variabilité dans les récompenses, indiquée par l'écart-type, bien que la tendance générale montre une augmentation continue des récompenses moyennes.

— Les performances globales sont stables, mais l'agent pourrait encore bénéficier de l'optimisation d'autres hyper-paramètres pour réduire la volatilité observée.

2.1.2 Conclusion

L'utilisation des hyper-paramètres de *Tianshou* a montré des résultats prometteurs, avec une amélioration régulière des récompenses moyennes et une certaine stabilité. Cependant, la courbe suggère que des ajustements supplémentaires des hyper-paramètres pourraient encore réduire les fluctuations et améliorer les performances globales de l'agent.

2.2 Références

— `https://github.com/thu-ml/tianshou/tree/master`

— `https://github.com/thu-ml/tianshou/blob/master/test/continuous/test_td3.py`